




SP-104 Active Learning System Images

Software Design Document (SDD)

CS 4850 – Fall 2025

September 21, 2025

			
Josh Smith Team Leader Developer	Elijah Merrill Developer	Noah Lane Documentation	Matthew Hall Documentation

Team Members:

Name	Role	Cell / Alt Email
Josh Smith (Team Lead)	Developer	706-414-2827 joshuasmith0515@gmail.com
Elijah Merrill	Developer	478-283-0811 elijahmerrill04@gmail.com
Matthew Hall	Documentation	678-873-8542 thematthewhall7@gmail.com
Noah Lane	Documentation	706-591-2312 noahlane142@gmail.com
Sharon Perry	Advisor	770-329-3895 sperry46@kennesaw.edu

Contents

1. Introduction.....	3
1.1 Purpose of SDD	3
2. Design Considerations	3
2.1. Assumptions and Dependencies.....	3
2.2. General Constraints	4
2.3. Development Methods	5
3. Architectural Strategies.....	6
4. System Architecture	7
5. Detailed System Design.....	7
5.1. Classification	7
5.2. Definition.....	8
5.3. Constraints	8
5.4. Resources.....	8
5.5. Interface/Exports	9
6. Glossary	9
7. Bibliography	10

1. Introduction

This document describes the overall design attributes that are planned to be implemented in the Active Learning System for labeling chest X-ray images. The system is being developed to reduce the manual annotation effort required for large-scale medical imaging datasets by integrating an active learning loop with uncertainty sampling. This document also explains more about how the requirements outlined in the Software Requirements Specification document translate into concrete design elements.

1.1 Purpose of SDD

The purpose of this Software Design Document is to define and communicate the system architecture, component design, data design, interface design, and design constraints of the Active Learning System for labeling chest X-ray images. The SDD is intended to serve as a reference throughout the development life cycle and will pertain to stay iterative, allowing for refinement and adjustments as new insights are gained during the implementation and testing. The primary audience for this document is the advisor, the development team, and any other individual curious about the design aspect of the system.

2. Design Considerations

2.1. Assumptions and Dependencies

The system will heavily rely on a given dataset to function properly, however, beyond this, there are relatively few dependencies. The dataset serves as the foundation for training, validating, and testing the active learning model, and its quality directly affects the system's accuracy and reliability. Without sufficient data diversity and size, the system may not generalize well to real-world cases. Otherwise, the system will mainly depend on the computing environment in which it is deployed. It will require reasonably up-to-date hardware, which will be discussed in the following section, to ensure smooth image processing and performance. Additionally, it is assumed that the user will operate the system on a modern operating system, as this will allow compatibility with the latest machine learning frameworks, imaging libraries, and security updates.

Another assumption is that the system may be used by a wide range of individuals, from those with extensive expertise in radiology and chest X-ray interpretation to those with little prior knowledge. This means the user interface, documentation, and usability features should be designed to accommodate varying levels of experience. Experts may use the system to validate results or support diagnostic tasks, while less experienced users may rely on it as a learning or decision-support tool.

Future modifications are not expected to be strictly necessary in the short term, given the current landscape of medical discoveries and available chest X-ray datasets. However, ongoing updates may be applied to improve performance, efficiency, and accuracy over time. Such updates could involve refining the active learning strategy, enhancing data preprocessing techniques, or optimizing the model architecture. In the long run, major modifications will likely only become necessary if significant new medical insights are discovered or if substantially more comprehensive datasets are released. In that case, the system may be adapted to incorporate new diagnostic markers, expanded categories of chest conditions, or even different types of medical imaging beyond chest X-rays. This flexibility ensures that while the system is stable and effective in its current form, it remains adaptable to future advancements.

2.2. General Constraints

The constraints imposed on the system are intended to ensure it remains manageable and always runs effectively. The environment in which the system operates should be based on reasonably modern architecture and hardware. While top-of-the-line hardware is not required, the system must be capable of processing thousands of images and provide adequate storage capacity. If the hardware cannot handle image processing efficiently, system throughput may decrease, and execution errors could occur.

The system is also constrained to use only publicly available datasets related to chest X-rays. It should not process any images that violate privacy or safety laws. Use is limited to chest X-rays and potentially other similar medical imaging tasks, but not in ways that interfere with institutions or businesses in the health and safety field. This active learning system is intended for baseline, routine use in identifying diseases and injuries observable in X-rays.

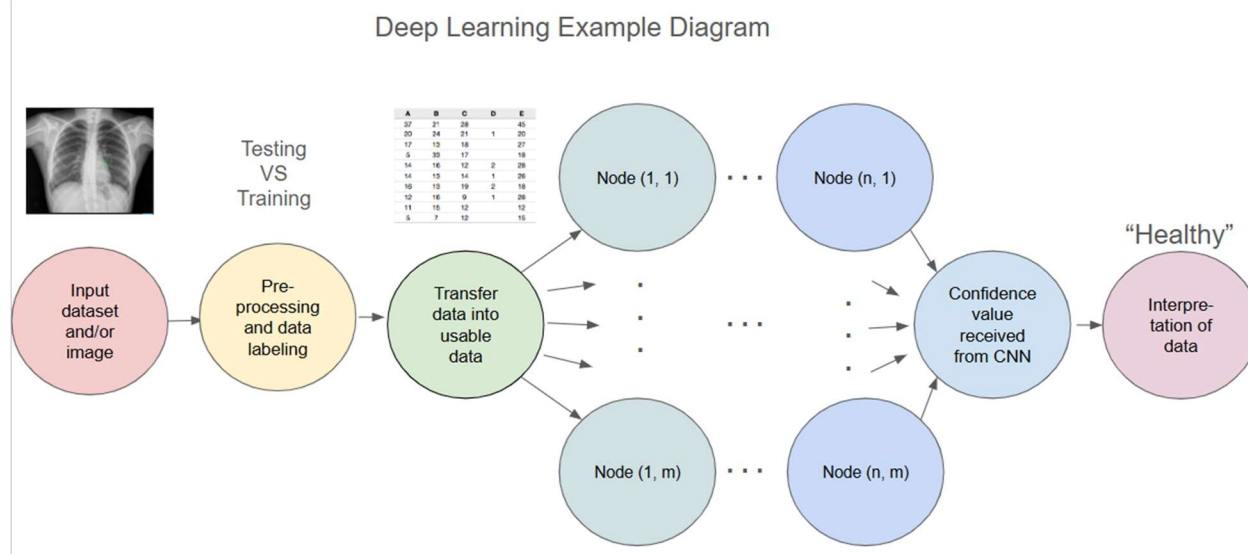
Although the datasets used are publicly available, meaning security is not the highest priority, basic measures should still be in place to ensure models and the system itself run efficiently and without data leaks. Image labels will be stored securely to prevent tampering. Overall, the system will not require extensive, enterprise-level security but will maintain essential protections for data and images.

Since the datasets will be acquired from online sources and stored locally, the system will not require significant network bandwidth once the data has been downloaded. However, the initial download process may involve large file transfers, as chest X-ray datasets often consist of tens of thousands of images, requiring both time and stable connectivity. After acquisition, all image data will be managed and accessed locally, reducing dependency on constant internet access. This ensures that the system can continue to function effectively even in environments with limited or unreliable network availability.

By operating primarily in a local environment, the system also reduces risks associated with transmitting sensitive medical data over networks, even if the datasets themselves are publicly

available. Local storage further allows for faster retrieval and processing of images during active learning cycles, improving system responsiveness. Updates to the dataset or model can still be applied periodically through controlled online access, but day-to-day functionality will remain self-contained. In this way, the system balances the efficiency of local processing with the flexibility of occasional online integration.

2.3. Development Methods

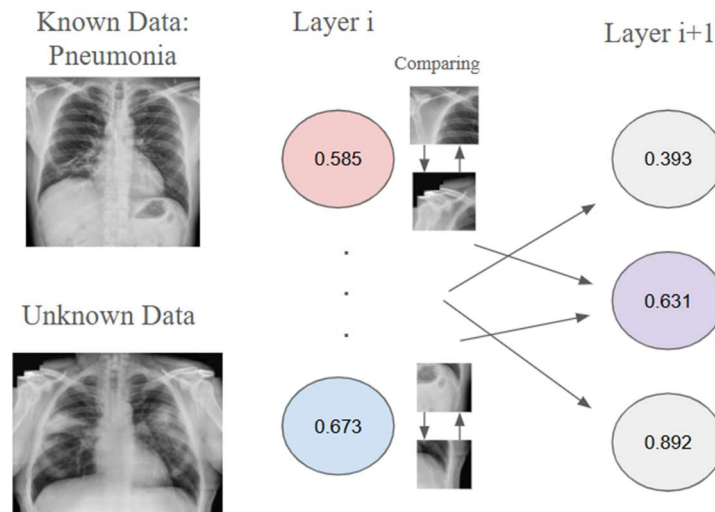


For this project, we must integrate the idea of using deep learning and Convolutional Neural Networks (CNNs) to create an AI that can understand the X-ray images it has been given. In the first section, we will need to implement a way for the program to accept the image (such as a PNG) and be able to implement a way for the ai to take data out of said image.

After implementing a method for storing the different images in the tables, we must create two datasets: one for known data and one for test data. We will store most of the images as known data in the table for the AI to compare information to. We will use data inside the images, such as the grey-scale rating, for each pixel. We store the data associated with the “known” images in a table that includes the patient's diagnosis, such as the diseases they may have, and the test data in a separate table without diagnostic information.

Once the table is created, the AI will then start to compare the unknown data individually with each section in the known section. This is the process that happens during the CNN part of the AI. A basic understanding of a CNN is that the CNN will break the work down into different layers, known as nodes. In each node, a value will be stored, such as an integer or a float value. In the first layer, the ai compares a singular pixel in the unknown data to similar pixels in known data and receives a value on its similarity. It will then continue with this process with each pixel in unknown data until it creates m nodes, where m in the number of pixels. The next layer will then take a grouping of pixels to compare known and unknown data. It will compare and create a

new value for each node in the second layer. This process will then be repeated, creating new groupings and nodes, n number of times.



Once the final layer is created using all previous data, the ai will come to a final value, which will show us its final verdict and confidence levels. We will then have the program interpret the data, printing out the result for us to see. We can then fix any confidence levels if needed or tweak some of the calculations to make the program more accurate.

3. Architectural Strategies

The system is structured as a set of modular components, including data ingestion, storage, preprocessing, model training, active learning, and the user interface. Data flows from raw images and metadata through preprocessing and baseline training into the active learning loop, which generates evaluation reports and stores model artifacts.

Python is chosen as the primary language because of its strong support for PyTorch and related libraries. While TensorFlow remains a possible alternative, PyTorch's flexibility and wide community adoption make it better suited for experimentation and integration. The model leverages pre-training to reduce computational costs and the need for large, labeled datasets. Images are stored in organized file systems, while metadata is managed in lightweight formats such as SQL databases, balancing scalability with simplicity.

The active learning loop relies on pool-based sampling that combines uncertainty and filtering diversity, with random sampling maintained as a baseline and ensemble methods recognized as more accurate but less efficient. The system also incorporates validation checks to ensure data integrity and robust error handling. Future extensions may include semi-supervised learning and explainability features to support potential clinical applications.

4. System Architecture

The architecture of the Active Learning System is designed to partition responsibilities into well-defined subsystems that would collectively enable efficient dataset management, model training, active learning iteration, and performance evaluation. At a high-level, the system is composed of five components: Dataset management, Model Training, Active Learning Engine, Evaluation and Visualization, and User Interface Layer.

The Dataset Management subsystem is responsible for various tasks like loading the NIH Chest X-Ray dataset and preprocessing images and metadata. This ensures that consistent, model-ready data is supplied to downstream modules. The Model Training subsystem handles both the baseline model initialization and the retraining process that occurs after each active learning iteration.

The Active Learning Engine serves as the core driver of the system. It applies to acquisition strategies such as uncertainty sampling to identify the most informative unlabeled samples for annotation. These selections are then routed back to the dataset management, where labels are added to the dataset.

The Evaluation and Visualization subsystem provides mechanisms for tracking performance after each iteration. It generates metrics like accuracy and recall while also visualizing learning curves that show how performance improves as more samples are labeled.

Finally, the User Interface Layer ties all components together by providing researchers with an accessible entry point into the system. Through the interface, users can load datasets, begin training, configure active learning loops, and load results.

5. Detailed System Design

5.1. Classification

- Dataset Management: Subsystem/module package with helper classes for I/O, metadata parsing, preprocessing, and dataset splits.
- Model Training: Subsystem service with trainer class and checkpointing utilities.
- Active Learning Engine: Subsystem service with pluggable acquisition strategies.
- Evaluation and Visualization: Subsystem analytics module with metric calculations and plotting utilities.
- User Interface Layer: Presentation layer: web UI (React/Flask/etc.) or interactive notebook widgets.

5.2. Definition

- **Dataset Management:** Responsible for reading the NIH ChestX-ray14 dataset, validating integrity, standardizing images (preprocessing), and maintaining labeled and Unlabeled pools. Produces deterministic train/val/test splits and exposes iterable data loaders.
- **Model Training:** Initializes the baseline CNN (to be trained on an optimal dataset to avoid bias with the chest X-ray dataset), configures loss, learning-rate scheduler, optimizer, and runs epochs over provided data loaders.
- **Active Learning Engine:** Coordinates select → label → update → retrain cycles. Computes acquisition scores on the unlabeled pool, selects a batch under a labeling budget, fetches labels (simulated from the dataset), promotes to the labeled pool, and triggers retraining.
- **Evaluation and Visualization:** Computes performance metrics. Produces learning curves, threshold sweeps, and per-round comparison reports (TBD).
- **User Interface Layer:** Provides the ability to select dataset path, view class histograms, launch baseline training, configure the acquisition strategy and batch size, monitor training, review metrics, and export results.

5.3. Constraints

- **Dataset Management:** Must be able to accurately process large amounts of images. Must comply with privacy and safety laws.
- **Model Training:** The model should only label the images of the dataset and not modify any images. System should be able to handle processing.
- **Active Learning Engine:** Should remain efficient even as the dataset scales, with selection algorithms optimized for large pools.
- **Evaluation and Visualization:** Must present results in an interpretable manner for both experts and non-experts, without overwhelming users.
- **User Interface Layer:** User should not inherently be able to interact with any of the innerworkings of the model and dataset. Must remain lightweight and responsive, even when handling large datasets.

5.4. Resources

- **Dataset Management:**
 - Disk: Image store + metadata CSVs.
 - Libraries: Image I/O (Pillow/OpenCV), table parsing (pandas), numeric (NumPy)
- **Model Training:**
 - Hardware: Decent GPU/CPU combo
 - Libraries: Pytorch
 - External: Optional experiment tracker (TensorBoard/MLflow)
- **Active Learning Engine:**
 - Pools: Access to Dataset Management APIs for labeled/unlabeled sets
 - Storage: Per-round ledger (TBD)
- **Evaluation and Visualization:**
 - Libraries: scikit-learn, matplotlib
 - Data: Access to predictions and targets

- User Interface Layer:
 - Static storage: resulting images/exports for download.
 - Concurrency: Queues for training jobs

5.5. Interface/Exports

Interface: `load_dataset()`, `split_data()`, `get_batch()`

Exports: labeled/unlabeled pools, standardized image tensors, and data statistics.

- Model Training:
 - Interface: `train_model()`, `evaluate_model()`, `save_checkpoint()`
 - Export: trained model weights, training logs, and performance metrics.
- Active Learning Engine:
 - Interface: `select_samples(strategy, k)`, `update_labels()`
 - Exports: List of selected images, updated label pools, and acquisition logs.
 - `k` would be an integer likely representing batch size of images for labeling per iteration.
 - Strategy would be a parameter telling the active learning engine how samples would be chosen.
- Evaluation and Visualization:
 - Interface: `compute_metrics()`, `plot_learn_curve()`
 - Exports: visualization like ROC/PR curves and JSON metrics
- User Interface Layer:
 - Interface: GUI buttons, notebook widgets
 - Exports: Interactive views (dashboards, tables) and downloadable (PDF, PNG)

6. Glossary

An ordered list of defined terms and concepts used throughout the document.

- Convolutional Neural Networks (CNNs) - a type of deep learning model, specifically a deep neural network architecture, designed to process data with a grid-like topology, such as images.
- Dataset - a collection of related sets of information that is composed of separate elements but can be manipulated as a unit by a computer.
- Uncertainty Sampling - an active learning strategy that selects unlabeled data points for human labeling that the current model is most uncertain about.
- Metadata - a set of data that describes and gives information about other data. “data about data”
- Training Data - the dataset, comprising labeled or unlabeled examples of text, images, audio, or other forms of data, used to train an algorithm to recognize patterns and make accurate predictions or perform specific tasks.

- Testing Data - In machine learning, testing data refers to a separate, independent portion of a dataset that is used for the final evaluation of a model after it has been fully trained and, if applicable, tuned using validation data.
- PyTorch - an open-source machine learning framework, primarily used for deep learning applications like computer vision and natural language processing, that provides a flexible and fast platform for building and training neural networks.
- Kaggle - a platform for data scientists and machine learning professionals that offers competitions, datasets, and a community to learn, share work, and collaborate on projects
- Institute of Electrical and Electronics Engineers (IEEE) – The company that develops industry standards, produces peer-reviewed publications, organizes conferences, and provides educational and professional development resources to its global community of engineers, scientists, and allied professionals

7. Bibliography

A list of referenced and/or related publications.

X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, “ChestX-Ray14: Data,” *Kaggle*, 2017. Available: <https://www.kaggle.com/datasets/nih-chest-xrays/data>

Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., ... & Ng, A. Y. (2019). “CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparisons” Available: <https://doi.org/10.1109/ICML.2019.00019>

National Institutes of Health Clinical Center. (2017). ChestXray-NIHCC dataset. Available: <https://nihcc.app.box.com/v/ChestXray-NIHCC>

GeeksforGeeks. (2023, November 29). Training data vs Testing data. Available: <https://www.geeksforgeeks.org/python/training-data-vs-testing-data/#what-is-testing-data>